Multidisciplinary Optimization in Decentralized Reinforcement Learning

Thanh Nguyen
Dept. of Computer and Information Science
Indiana University - Purdue University - Indianapolis
Indiana, United States
thamnguy@imail.iu.edu

Snehasis Mukhopadhyay
Dept. of Computer and Information Science
Indiana University - Purdue University - Indianapolis
Indiana, United States
smukhopa@cs.iupui.edu

Abstract— **Multidisciplinary Optimization (MDO) is one of the most popular techniques in aerospace engineering, where the system is complex and includes the knowledge from multiple fields. However, according to the best of our knowledge, MDO has not been widely applied in decentralized reinforcement learning (RL) due to the 'unknown' nature of the RL problems. In this work, we apply the MDO in decentralized RL. In our MDO design, each learning agent uses system identification to closely approximate the environment and tackle the 'unknown' nature of the RL. Then, the agents apply the MDO principles to compute the control solution using Monte Carlo and Markov Decision Process techniques. We examined two options of MDO designs: the multidisciplinary feasible and the individual discipline feasible options, which are suitable for multi-agent learning. Our results show that the MDO individual discipline feasible option could successfully learn how to control the system. The MDO approach shows better performance than the completely decentralization and centralization approaches.**

Keywords—multidisciplinary optimization; reinforcement learning; nonlinear system; Markov decision process

I. INTRODUCTION

Since 1970s, decentralized computing, which is also referred as distributed computing or multi-agent systems, has been evolving and rapidly and becoming one of the most attractive area in intelligent systems, including reinforcement learning (RL). Decentralization is, perhaps, the most feasible approach when the learning problems are in high dimension with large amount of data [1], or when the problems are complex and naturally distributed by geographical locations or by disciplinary [2, 3]. With decentralization, the learning algorithms operate on less number of state variables and are less susceptible to uncertain system parameters [4]. In addition, decentralization makes the system more adaptive to structural changes than the corresponding centralized systems [5]. Another benefit of decentralization is that if one agent fails in learning, the other agents could compensate for it in the overall learning problem resulting in only graceful degradation of performance [3]. Examples of decentralized RL could be found in many applications, such as networking [6], robotics [7] and intelligent control [8].

There are four major challenges in decentralized RL. First, the central issue in decentralization is the question of when and how to interact with whom [1]. Second, due to the interaction among the learning agents, each learning agent faces another challenge in defining a learning goal: should the agent only focus on its own or should it incorporate the other agents' goals in its learning goal [3]? Third, in RL, the decentralized learning has an additional difficulty due to the unknown environment and lack of precise feedback on its learning performance [9]. This challenge could be tackled by system identification [10-12]. Forth, finding the optimal solution for the learning problem may not be easy because the closed-form solution for the Hamilton-Jacobi-Bellman (HJB) equation, is unknown in general cases. Therefore, researchers have been focusing on approximation methods to tackle nonlinear HJB equation problem such as [13-16].

Mathematically, **m**ulti**d**isciplinary **o**ptimization (MDO) [17, 18], which has been intensively researched and applied in aerospace and engineering, could be a promising approach to tackle the first two challenges in decentralized RL. In MDO, the computational agents are well-defined and decomposed according to the domain-knowledge of each discipline in a jointed optimization problem. A typical example of MDO could be found in [19]. In the formulation step, there are two typical options. First in the multidisciplinary feasible (MF) option (which may also called 'boarder sense' option), each computational agent incorporate the information from the other agents in its own optimization function [19]. In addition, each agent only uses the optimization constrains from its own discipline. In this option, because the each agent includes the information from other agents in the optimization objective, the agent tends to approach closer to the global optimal solution. Second, in the individual discipline feasible (IDF) option (which may also called the 'selfish' option), each computational agent only aims to optimize its own optimization function and uses the other agents' information as constrains. In this option, the agents tends to seek for local optimization; and the constrains from other agents will drive the local solutions to the global solution. More literature details about MDO approach could be found in [20-22].

However, according to our best knowledge, MDO has not been widely applied in decentralized RL. In our opinion, there are two factors limiting the capability of the MDO in RL. First, the unknown nature and long-term goal of the RL problems implies that we could not get the optimization function in closed-form $J(\mathbf{x})$, where J denotes the optimization objectives in RL and $\mathbf{x}$ denotes the variables in the RL problem. Since MDO

This is the author's manuscript of the article published in final edited form as:
Nguyen, T., & Mukhopadhyay, S. (2017). Multidisciplinary Optimization in Decentralized Reinforcement Learning. In 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 779–784). https://doi.org/10.1109/ICMLA.2017.00-63

techniques rely on numerical methods - especially the gradient methods - to solve the optimization problem, without a closed-form of $J(\mathbf{x})$ [17], most of the state-of-the-art MDO techniques are inapplicable. Second, even when the unknown nature of the RL problems could be solved by system identification methods [23], the closed-form solution for many of the Hamilton-Jacobi-Bellman equation, is very difficult to find for the MDO to compute the control unit.

In this paper, we propose applying the MDO idea in solving decentralized RL problems and demonstrates the capability of the MDO approach in several learning adaptive control toy examples. In our approach, we use nonlinear system identification to approximate the unknown environment in the RL problem. Each agent setups the Markov Decision Process (MDP) by system identification to compute the action/control solution, which has been shown in [24]. We compare the control solutions computed using both the MF and the IDF options. We also examine the exchanged information among the agents. We focus on the question: how much the learning performance loss when the exchanged information among the agents is simplified. Then, we compare the learning performance of the MDO approach with the strictly decentralized approach, the selectively decentralized approach [24] and the centralized approach and discuss the advantage of the MDO in our learning examples.

II. Methods

A. Problem statements

1) The learning adaptive control problem

In this paper, we focus on discrete time-invariant and continuous systems in the general format

$$\mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t)) \tag{1}$$

where $\mathbf{x} \in \Re^n$ stands for the n dimensional bounded state vector, $\mathbf{u} \in \Re^m$ stands for the m dimensional bounded control unit, t stands for the iteration or time and $f : \Re^n \times \Re^m \to \Re^n$ is a continuously differentiable unknown function. Here, the symmetric boundaries $[-\chi, \chi]$ and $[-\mu, \mu]$ for all components of $\mathbf{x}$ and $\mathbf{u}$ are known. We also assume that the agent fully observe $\mathbf{x}(t)$ in every iteration. Given p: $\Re^n \to \Re$ and q: $\Re^m \to \Re$ as the two continuously semi-definite negative and differentiable reward functions with the following properties

$$p(\mathbf{x}_1) \le p(\mathbf{x}_2) \Leftrightarrow \|\mathbf{x}_1\| \ge \|\mathbf{x}_2\| \text{ and } p(\mathbf{0}) = 0 \tag{2}$$

$$q(\mathbf{u}_1) \le q(\mathbf{u}_2) \Leftrightarrow \|\mathbf{u}_1\| \ge \|\mathbf{u}_2\| \text{ and } q(\mathbf{0}) = 0 \tag{3}$$

where $\|\mathbf{x}\|$ denotes the second norm of $\mathbf{x}$. The main objective is to learn the control unit $\mathbf{u} = u(\mathbf{x})$ such that

$$\mathbf{x}(t) \to 0, \mathbf{u}(t) \to 0 \text{ as } t \to \infty \tag{4}$$

For optimal control, we convert the objective in (4) into the optimization objective with discount factor $0 < \gamma \to 1$

$$J(\mathbf{x}(0)) = \sum_{t=0}^{\infty} \gamma^t \left(p(\mathbf{x}(t)) + q(\mathbf{u}(t))\right) \tag{5}$$

The equations (1-5) defines a HJB equation.

2) The system identification problem statements

Since we use MDP, a model-based method, to compute $\mathbf{u}$, we need an approximation of the environment. The objective is to find the approximated nonlinear function $\hat{f}$ such that with the predicted state vector

$$\hat{\mathbf{x}}(t+1) = \hat{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{6}$$

the identification error

$$e(t) = \|\mathbf{x}(t) - \hat{\mathbf{x}}(t)\| \tag{7}$$

approaches 0 as $t \to \infty$.

3) Assumptions about decentralization and MDO

The assumptions for applying MDO is as follow. First, the system (1) could be decoupled into multiple subsystems by domain knowledge. Each subsystem corresponds to one learning agent. Second, each agent knows precisely which components of $\mathbf{x}$ and $\mathbf{u}$ affecting its learning performance. Third, each agent's control unit does not directly and instantly affect the $\mathbf{x}$ components of the other agents. In the other words,

$$\mathbf{x}(t+1) = \begin{bmatrix} \mathbf{x}_1(t+1) \\ \mathbf{x}_2(t+1) \\ \vdots \\ \mathbf{x}_k(t+1) \end{bmatrix} = f(\mathbf{x}(t), \mathbf{u}(t)) = \begin{bmatrix} f_1(\mathbf{x}_1(t), C_1(\mathbf{x}_{i\neq 1}(t)), \mathbf{u}_1(t)) \\ f_2(\mathbf{x}_2(t), C_2(\mathbf{x}_{i\neq 2}(t)), \mathbf{u}_2(t)) \\ \vdots \\ f_k(\mathbf{x}_k(t), C_k(\mathbf{x}_{i\neq k}(t)), \mathbf{u}_k(t)) \end{bmatrix} \tag{8}$$

where k is the number of learning agents, i stands for agent index and C is the bounded communication function, which is known, among the agents. Forth, for all agents, C has the following property

$$\| C_j(\mathbf{x}_{i\neq j}(t)) \| \ge \| C_j(\mathbf{x'}_{i\neq j}(t)) \| \Leftrightarrow \| \mathbf{x}_{i\neq j}(t) \| \ge \| \mathbf{x'}_{i\neq j}(t) \| \tag{9}$$

where $\mathbf{x'}$ stands for another state vector. In general, for agent j, $C_j(\mathbf{x}_{i\neq j}(t))$ should be simpler than $\mathbf{x}_{i\neq j}(t)$, such as having less dimension, to reduce the computational cost. Fifth, the agents can freely exchange their state information. In the most complicated communication, one agent can send the exact state information to the other agents. From these assumptions, we can rewrite the identification problem as

$$\mathbf{x}(t+1) = \begin{bmatrix} \mathbf{x}_1(t+1) \\ \mathbf{x}_2(t+1) \\ \vdots \\ \mathbf{x}_k(t+1) \end{bmatrix} = f(\mathbf{x}(t), \mathbf{u}(t)) = \begin{bmatrix} f_1(\mathbf{x}_1(t), C_1(\mathbf{x}_{i\neq 1}(t)), \mathbf{u}_1(t)) \\ f_2(\mathbf{x}_2(t), C_2(\mathbf{x}_{i\neq 2}(t)), \mathbf{u}_2(t)) \\ \vdots \\ f_k(\mathbf{x}_k(t), C_k(\mathbf{x}_{i\neq k}(t)), \mathbf{u}_k(t)) \end{bmatrix} \tag{10}$$

The identification in (8) allows solving the HJB equation (1)-(5) by both MDO's MF option and IDF option. For the MF option, each learning agent j has optimization function according to (5)

$$J(\mathbf{x}_j(0)) = \sum_{t=0}^{\infty} \gamma^t \left(p_j(\mathbf{x}_j(t), C_j(\mathbf{x}_{i\neq j}(t))) + q_j(\mathbf{u}_j(t))\right) \tag{11}$$

Therefore, the MDP for agent j in the MF option has the form $[\mathbf{x}_j, C_j(\mathbf{x}_{i\neq j})] \times [\mathbf{u}_j] \rightarrow [\mathbf{x}_j, C_j(\mathbf{x}_{i\neq j})]$. For the IDF option, each agent optimizes

$$J(\mathbf{x}_j(0)) = \sum_{t=0}^{\infty} \gamma^t \left(p_j\left(\mathbf{x}_j(t)\right) + q_j\left(\mathbf{u}_j(t)\right)\right) \quad (12)$$

In this case, the MDP has the form $[\mathbf{x}_j] \times [\mathbf{u}_j] \rightarrow [\mathbf{x}_j]$. Because each MDP in IDF option does not cover the entire knowledge gained by identification, each agent may have multiple MDPs depending on the $C_j(\mathbf{x}_{i\neq j})$. Here, we have the sixth assumption: the agents' reward functions p_j and q_j in (11) and (12) have the same properties to (2) and (3).

B. Identification for nonlinear unknown system

We use the three-layer feedforward neural network to approximate f as $\hat{f}$ in nonlinear system identification. Theoretical foundation and application of neural network as such universal function approximators in control systems can be found in [13, 25, 26]. We use the backpropagation learning algorithm for neural networks [27]. For the MDO's MF option, $\{\mathbf{x}_j(t), C_j(\mathbf{x}_{i\neq j}(t)), \mathbf{u}_j(t)\}$ is presented at the input layer, $\{\hat{\mathbf{x}}_j(t+1), \hat{C}(\mathbf{x}_{i\neq j}(t+1))\}$ is computed at the output layer of the neural network, and $\{\mathbf{x}_j(t+1), C_j(\mathbf{x}_{i\neq j}(t+1))\}$ is the target. In the other hands, for the IDF option, $\{\mathbf{x}_j(t), C_j(\mathbf{x}_{i\neq j}(t)), \mathbf{u}_j(t)\}$ is presented at the input layer, $\{\hat{\mathbf{x}}_j(t+1)\}$ is computed at the output layer of the neural network, and $\{\mathbf{x}_j(t+1)\}$ is the target. Without MDO for the comparative approaches, the neural network could be set up as in [24].

C. Discretization and MDP construction

In this paper, we further develop the discretization method and MDP construction from the simulation in [24]. Due to the limitation of space, we omit the theoretical assessment of the discrete MDP method to approximate the HJB equation's solution (1)-(5). In this section, first, we would briefly present the discrete MDP method to solve (1)-(5) by the centralized and completely decentralized approach, which does not involve any MDO principles. Second, we would present the modification of the discrete MDP method to apply MDO principles in both the MF and the IDF options.

1) Discrete MDP method for the centralized and completely decentralized approach

a) Discretize the state and action vectors

The details in this step could be found in [28]. Here, we briefly summarize the parameters for discretization as follow. Let M be the number of intervals in each dimension of $\mathbf{x}$ and $\mathbf{u}$ for which we uniformly divide the dimension into small grids. Therefore, the entire state space is divided into M^n small hyper cubes with edge $\theta_x = 2\chi/M$. The control space is divided into M^m small hyper cubes with edge $\theta_u = 2\mu/M$. Mathematically, the discretization process is described by the following formulas

$$\mathbf{x}[d] \rightarrow \theta_x + \chi/M \; and \; \mathbf{x}[d] \in [\theta_x, \theta_x + 2\chi/M) \quad (13)$$

$$\mathbf{u}[d] \rightarrow \theta_u + \mu/M \; and \; \mathbf{u}[d] \in [\theta_u, \theta_u + 2\mu/M) \quad (14)$$

where d is the dimension index, $\theta_x \in \{-\chi, -\chi + 2\chi/M, -\chi + 4\chi/M, ..., \chi - 2\chi/M\}$ and $\theta_u \in \{-\mu, -\mu + 2\mu/M, -\mu + 4\mu/M, ..., \mu - 2\mu/M\}$, which are the 'left' boundaries in the hyper cubes. We denote $\mathbf{x}_{dis}$ and $\mathbf{u}_{dis}$ as the discrete space and control vector of $\mathbf{x}$ and $\mathbf{u}$, correspondingly. We also denote $(\mathbf{x}_{dis})$ and $(\mathbf{x}_{dis})$ as the set/supspace of points $\mathbf{x}$ and $\mathbf{u}$ whose discrete forms are $\mathbf{x}_{dis}$ and $\mathbf{u}_{dis}$, correspondingly.

b) Setup the probabilistic transition function for the MDP

The MDP requires the probabilistic transition function as a matrix of $P(\mathbf{x'}_{dis} \mid \mathbf{x}_{dis}, \mathbf{u}_{dis})$, which is the probability of reaching discrete state $\mathbf{x'}_{dis}$ when executing action $\mathbf{u}_{dis}$ at state $\mathbf{x}_{dis}$. Since the neural network could train $\hat{f}$ closed to the unknown f at any precision with sufficient training points, we can apply the Monte Carlo method [29] to approximate $P(\mathbf{x'}_{dis} \mid \mathbf{x}_{dis}, \mathbf{u}_{dis})$ as follow.

- Generate a large number of S points $(\mathbf{x}, \mathbf{u})$ following the uniform distribution in $(\mathbf{x}_{dis}) \times (\mathbf{u}_{dis})$.

- Count the number of points S_1 such that $\hat{f}(\mathbf{x}, \mathbf{u}) \in (\mathbf{x'}_{dis})$.

- Then $S_1/S \rightarrow P(\mathbf{x'}_{dis} \mid \mathbf{x}_{dis}, \mathbf{u}_{dis})$ as $S \rightarrow \infty$.

2) Discrete MDP method for the MDO approach

The central theme in this section is the discretization of communication $C_j(\mathbf{x}_{i\neq j})$. Let us call n_j, m_j and $n_{i\neq j}$ be the dimensionality of the $\mathbf{x}_j$, $\mathbf{u}_j$ and $C_j(\mathbf{x}_{i\neq j})$ in agent j. Since the system (1) and communication function is bounded, each dimension of $C_j(\mathbf{x}_{i\neq j})$ is bounded by $[-\lambda, \lambda]$. Since the communication sent by agent i to agent j should not be more complex than $\mathbf{x}_i$, we discretize each dimension of $C(\mathbf{x}_{i\neq j})$ into $N \leq M$ grids using the same method described in (13) and (14). We denote the discrete from of $C_j(\mathbf{x}_{i\neq j})$ by $C_j(\mathbf{x}_{i\neq j})_{dis}$.

For the MDO-MF option, since the identified model is in the form $\{\mathbf{x}_j(t), C_j(\mathbf{x}_{i\neq j}(t)), \mathbf{u}_j(t)\} \rightarrow \{\mathbf{x}_j(t+1), C_j(\mathbf{x}_{i\neq j}(t+1))\}$, each agent j has one MDP model with dimensionality $(M^{n_j} \times N^{n_{i\neq j}}) \times M^{m_j} \times (M^{n_j} \times N^{n_{i\neq j}})$. The transition probability for the MDP could be setup similarly to the centralized approach, except the Mote Carlo sampling should be done on $(\mathbf{x}_{jdis}) \times (C_j(\mathbf{x}_{i\neq j})_{dis}) \times (\mathbf{u}_{jdis})$ in the whole space.

For the MDO-IDF option, since the identified model is in the form $\{\mathbf{x}_j(t), C_j(\mathbf{x}_{i\neq j}(t)), \mathbf{u}_j(t)\} \rightarrow \{\mathbf{x}_j(t+1)\}$, each MDP in this option will have dimensionality $(M^{n_j}) \times M^{m_j} \times (M^{n_j})$. By rewriting the identification as

$$\mathbf{x}_j(t+1) \approx \hat{f}\left(\mathbf{x}_j(t), C_j\left(\mathbf{x}_{i\neq j}(t)\right), \mathbf{u}_j(t)\right) =$$

$$\begin{cases} \hat{f}_{C_j\left(\mathbf{x}_{i\neq j}(t)\right)=\mathbf{c}_1}\left(\mathbf{x}_j(t), \mathbf{u}_j(t)\right) \\ \hat{f}_{C_j\left(\mathbf{x}_{i\neq j}(t)\right)=\mathbf{c}_2}\left(\mathbf{x}_j(t), \mathbf{u}_j(t)\right) \\ \vdots \\ \hat{f}_{C_j\left(\mathbf{x}_{i\neq j}(t)\right)=\mathbf{c}_{...}}\left(\mathbf{x}_j(t), \mathbf{u}_j(t)\right) \end{cases} \quad (15)$$

where $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}$… are possible the discrete values of $C_j(\mathbf{x}_{i\neq j}(t))$, it is easy to see that each agent j has $N^{n_{i\neq j}}$ MDP models, which could be indexed. When the agent receives $\{\mathbf{x}_j(t), C_j(\mathbf{x}_{i\neq j}(t))\}$, it looks up the index of $C_j(\mathbf{x}_{i\neq j}(t))$ and chooses the corresponding MDP to compute $\mathbf{u}(t)$.

D. The overall design of the learning adaptive control problem

The pseudo code for each agent is as follow. With a window size Ω deciding how frequent we call system identification:

Initialize:

$\hat{f}$ neural network with random weights.

Predefine the discretization parameters as in (13-14).

Construct the MDPs.

$u_j(\{\mathbf{x}_j, C_j(\mathbf{x}_{i\neq j})\})$ (or $u_j([\mathbf{x}_j])$): solution of the MDPs by policy iteration for the MF (or IDF) options.

For t from 2 to the maximum number of iterations

Receive and discretize $\{\mathbf{x}(t), C_j(\mathbf{x}_{i\neq j}(t))\}$ as in (13-14).

Obtain $\mathbf{u}(t) = u(\{\mathbf{x}_j(t), C_j(\mathbf{x}_{i\neq j}(t))\})$ (or $\mathbf{u}(t) = u(\mathbf{x}(t))$) according to the MDP policy.

Add $\{\mathbf{x}(t\text{-}1), C_j(\mathbf{x}_{i\neq j}(t\text{-}1)), \mathbf{u}(t\text{-}1)\}$ and $\{\mathbf{x}_j(t), C_j(\mathbf{x}_{i\neq j}(t))\}$ into the training set for future neural network training set.

if t % $\Omega = 0$

Retrain $\hat{f}$.

Reconstruct the MDPs.

Recompute $u_j(\{\mathbf{x}_j, C_j(\mathbf{x}_{i\neq j})\})$ (or $u_j([\mathbf{x}_j])$) by policy iteration.

Clear the training set of the neural network.

end if

end for

III. Simulation results

In this paper, we setup toy sinusoidal systems to demonstrate the capability of the MDO approach in RL. These systems has the form

$$\mathbf{x}(t) = \sin(\mathbf{A}\mathbf{x}(t\text{-}1)+\mathbf{u}(t\text{-}1)) \quad (20)$$

where $\mathbf{A}$ are 3×3 random Markov matrices. The boundary for $\mathbf{x}$ and u is $\chi = \mu = 0.35$. The setup for matrix $\mathbf{A}$ and definition of the vector sin function could be found in [28]. Briefly, σ is the ratio between the sum of non-diagonal entries in $\mathbf{A}$ and the sum of all entries in $\mathbf{A}$. In our simulations σ ranges from 0 to 0.3. With $\sigma = 0$, $\mathbf{A}$ becomes the identity matrix or the systems are completely decouple. The systems are more coupled when σ increases. In this design, we assign three learning agents such that each agent is responsible for one dimension of $\mathbf{x}$ and $\mathbf{u}$. The initial state $\mathbf{x}(0)$ is a vector of $\mathbf{0.2}$. For equation (2) and (3), we choose $p(\mathbf{x}) = -\|\mathbf{x}\|^2$ and $q(\mathbf{u}) = -\|\mathbf{u}\|^2$ for both the centralized approach and the learning agents in the decentralized / MDO approaches. For equation (5), $\gamma = 0.9$

For identification, each agent has neural networks of 50 hidden layers. The input and output layer is as in section II.2. We use the window size of $\Omega = 50$ (figure 1). In each training round, the training data set is reused at most 1000 times (epoch)

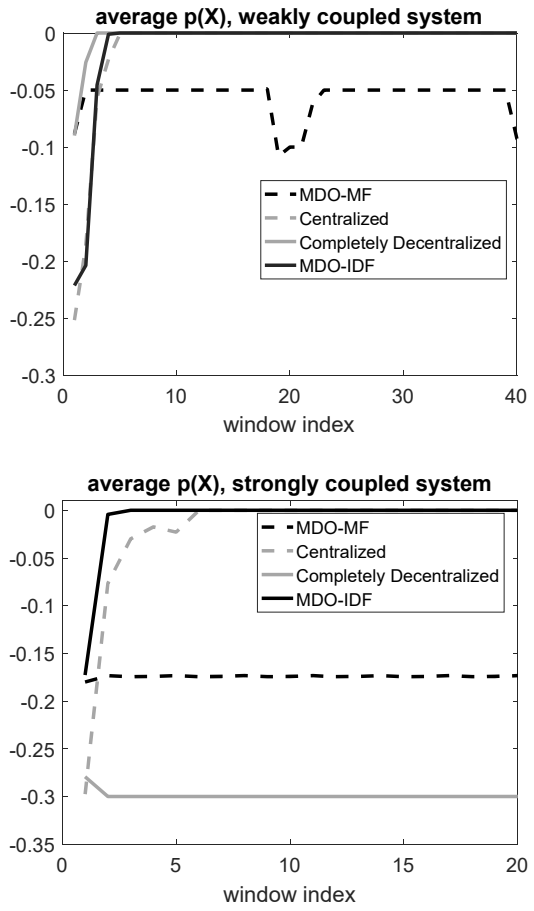

Fig.1. learning performance in $p(\mathbf{x})$ of the MDO approaches in weakly coupled system ($\sigma = 0.05$) and strongly coupled system ($\sigma = 0.3$).

[31] to improve identification. The maximum number of iterations t is 5000.

For the MDO discretization, each agent can fully send its state information to the other agents, which means $C_j(\mathbf{x}_{i\neq j}) = \mathbf{x}_{i\neq j}$ $\forall i, j$. Each agent j divides its $\mathbf{x}_j$ and $\mathbf{u}_j$ dimension into $M = 7$ grids (equation (13)-(14)). Therefore, the grid size in each dimension is 0.1. For the discretization of $C_j(\mathbf{x}_{i\neq j})$, we setup two scenarios:

- When the agent j uses the external information fully (with the same resolution) as it does for the internal $\mathbf{x}_j$ and $\mathbf{u}_j$, each dimension of $C_j(\mathbf{x}_{i\neq j})$ is divided into $N = M = 7$ grids. Therefore, in the MF option, each agent j has one MDP model of size $(7\times7^2) \times 7 \times (7\times7^2)$. In the IDF option, each agent j has 7^2 MDP models of size 7×7×7.

- When the agent j uses the external information less than (with less resolution) it does for the internal $\mathbf{x}_j$ and $\mathbf{u}_j$, each dimension of $C_j(\mathbf{x}_{i\neq j})$ is divided into $N = 5$ and $N = 3$ grids. Therefore, in the MF option, each agent j has one MDP model of size $(7\times5^2) \times 7 \times (7\times5^2)$ and $(7\times3^2) \times 7 \times (7\times3^2)$. In the IDF option, each agent j has 5^2 and 3^2 MDP models of size 7×7×7.

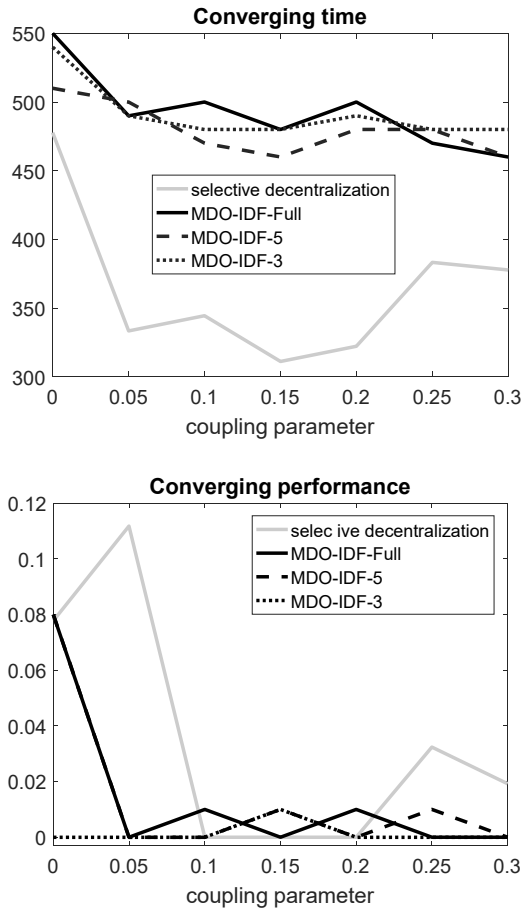

Fig.2. Converging time of the MDO-IDF approach with full and less resolution

For the completely decentralized, selectively decentralized and the centralized approach, each agent also divides each dimension of $\mathbf{x}$ and $\mathbf{u}$ into $M = 7$ grids.

Figures 1 compare the learning performance among the MDO, the complete decentralization and the centralization approaches. Due to the lack of space, we only show the result for the state vector $\mathbf{x}$ (refer to (4)). Overall, the MDO-IDF outperforms both the centralization and the complete decentralization approach. When the centralization is expected to have advantage in strongly coupled systems, the MDO-IDF minimize $p(\mathbf{x})$ to 0, or $\mathbf{x} \to 0$. $\mathbf{x}$ also converges to 0 at the similar speed in the centralized approach. In the other hand, the MDO-MF is better in learning when the system is strongly decoupled.

In figure 2, we show that the learning performance of the MDO-IDF approach are similar when the communication $C_j(\mathbf{x}_{i\neq j})$ is used with less resolution, compared with the selective decentralization approach. In this figure, we denote MDO-IDF-Full as applying the IDF approach with $M = N = 7$, MDO-IDF-5 as $M = 7$ / $N = 5$ and MDO-IDF-3 as $M = 7$ / $N = 3$. Here, we address the performance of these approach by converging time, which is defined as $\Omega \times w$, in which w is the window such that the average $p(\mathbf{x}) + q(\mathbf{u})$ at window w and w-1 are less than 0.001, or when the change of average $p(\mathbf{x}) + q(\mathbf{u})$ at w and w-1 is less than 0.001. Overall, for converging time, the selective decentralization converging time is still better than the MDO-IDF approach. However, at the converging window, the average $p(\mathbf{x}) + q(\mathbf{u})$ is closer to 0 in the MDO-IDF approach.

IV. Discussions

The paper highlight three key points. First, as in many other works, system identification could be used to tackle the unknown challenge in RL [10-12]. Second, the MDP is used to tackle the unknown close-form solution, which is one of the major factors limiting the application of MDO in RL [24]. Third, the design of agents' learning goals: 'boarder sense' or 'selfish' could significantly decide the result of applying MDO in RL. With the right design of learning goal, simplifying the exchanged information among the agents may not degrade learning performance.

In this paper, we show the capability of MDO approaches in decentralized reinforcement learning. It is clear that from our experiments, the MDO-IDF option could successfully stabilize the system in control-and-stabilize learning problem; meanwhile, the MDO-MF option is not always successful. Mathematically, this fact could be explained by the divergence of $f(\mathbf{x}_j, C_j(\mathbf{x}_{i\neq j}))$. In the MF option, $C_j(\mathbf{x}_{i\neq j})$ participates in the MDP construction in such a way that the Monte Carlo randomize samples of $C_j(\mathbf{x}_{i\neq j})$ through its range. When the system is more coupled, $f(\mathbf{x}_j, \mathbf{u}_j, C_j(\mathbf{x}_{i\neq j}))$ will be more diverge, even when $\mathbf{x}_j$ is already in the stable region. Therefore, the conditional probability $P(\mathbf{x}_j(t+1) \mid \mathbf{x}_j(t)=\mathbf{0}, \mathbf{u}_j(t)=\mathbf{0})$ such that both $\mathbf{x}_j(t+1)$ and $\mathbf{x}_j(t)$ are in the stable region is less when the system is more coupled. Thus, the utility value of $\mathbf{x}$ in the stable region become less; therefore, the agents see less 'motivation' to stabilize. In the individual feasible option, the MDP is constructed by $f_{Cj(\mathbf{x}i\neq j)}(\mathbf{x}_j, \mathbf{u}_j)$ where $C_j(\mathbf{x}_{i\neq j})$ is fixed. Since the simulation examples satisfy that all of the agent could stabilize their state to 0 together, we do not see the diverse of $f_{Cj(\mathbf{x}i\neq j)}(\mathbf{x}_j, \mathbf{u}_j)$ when both $\mathbf{x}_j$ and $C_j(\mathbf{x}_{i\neq j})$ are in stable regions. In the other words, from philosophical perspective, in a cooperative task, trusting the behavior of the collaborators often lead to better results than doubting the incompetence or error from the collaborators.

Compared to the selective decentralization approach [24], which is also showed to outperform both the centralization and the complete decentralization, the MDO-IDF offer a complimentary technique to tackle the communication among the learning agents. In the selective decentralization approach, there exist a central cooperative unit deciding which communication scheme to be used for the agents to make decision. In the other words, selective decentralization is about model switching and the communication among all of the agents are not always free. In the MDO-IDF, the agents could freely send the state information to the others. In addition, each agent is responsible for its own communication: how to use the communication to compute the best action. Both of these methods show better performance than huge centralization and blind (completely) decentralization approach in many cases of system decoupling.

Three are several limitations in this paper. The first limitation in this paper is the choice of communicating function.

Here, we use $C_j(\mathbf{x}_{i\neq j}) = \mathbf{x}_{i\neq j}$. In more specific problems, there may exist better communication function allowing less resolution for discretization of $C_j(\mathbf{x}_{i\neq j})$ but without performance loss. Or, $C_j(\mathbf{x}_{i\neq j})$ could be designed with dimensional reduction to speed up computation. Second, this paper uses MDP to compute the control unit. This technique is known to consume high amount of memory to store the state-transition probability. Other approaches with less memory consumption, such as adaptive dynamic programming [32, 33] could be applied in computing the control unit. Third, the neural network in this paper is used limitedly for system identification. Forth, due to the lack of space, we could not include theoretical analysis for the convergence of the MDO. In the future work, we could utilize the neural network for more tasks, as showed in [34-36], with more comprehensive theoretical analysis and experiment on benchmark dataset.

ACKNOWLEDGMENT

The research presented in this paper was supported by a National Science Foundation grant (No. ECCS-1407925).

REFERENCES

[1] Weiss, G., Multiagent systems: a modern approach to distributed artificial intelligence. 1999: MIT press.

[2] Buşoniu, L., R. Babuška, and B. De Schutter, Multi-agent reinforcement learning: An overview, in Innovations in multi-agent systems and applications-1. 2010, Springer. p. 183-221.

[3] Busoniu, L., R. Babuska, and B. De Schutter, A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2), 2008, 2008.

[4] Ioannou, P.A., Decentralized adaptive control of interconnected systems. Automatic Control, IEEE Transactions on, 1986. 31(4): p. 291-298.

[5] Shi, L. and S.K. Singh, Decentralized adaptive controller design for large-scale systems with higher order interconnections. Automatic Control, IEEE Transactions on, 1992. 37(8): p. 1106-1118.

[6] Al-Rawi, H.A., M.A. Ng, and K.-L.A. Yau, Application of reinforcement learning to routing in distributed wireless networks: a review. Artificial Intelligence Review, 2015. 43(3): p. 381-416.

[7] de Bruin, T., et al. Improved deep reinforcement learning for robotics through distribution-based experience retention. in Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on. 2016. IEEE.

[8] Luo, B., H.-N. Wu, and H.-X. Li, Data-based suboptimal neuro-control design with reinforcement learning for dissipative spatially distributed processes. Industrial & Engineering Chemistry Research, 2014. 53(19): p. 8106-8119.

[9] Hinton, G.E. and T.J. Sejnowski, Unsupervised learning: foundations of neural computation. 1999: MIT press.

[10] 1Wang, J.-S. and Y.-P. Chen, A fully automated recurrent neural network for unknown dynamic system identification and control. Circuits and Systems I: Regular Papers, IEEE Transactions on, 2006. 53(6): p. 1363-1372.

[11] Xiao-Qun, W. and L. Jun-An, Parameter identification and backstepping control of uncertain Lü system. Chaos, Solitons & Fractals, 2003. 18(4): p. 721-729.

[12] Narendra, K.S. and K. Parthasarathy, Identification and control of dynamical systems using neural networks. Neural Networks, IEEE Transactions on, 1990. 1(1): p. 4-27.

[13] Abu-Khalaf, M. and F.L. Lewis, Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. Automatica, 2005. 41(5): p. 779-791.

[14] Saridis, G.N. and C.-S.G. Lee, An approximation theory of optimal control for trainable manipulators. Systems, Man and Cybernetics, IEEE Transactions on, 1979. 9(3): p. 152-159.

[15] Beard, R.W., G.N. Saridis, and J.T. Wen, Galerkin approximations of the generalized Hamilton-Jacobi-Bellman equation. Automatica, 1997. 33(12): p. 2159-2177.

[16] Huang, C.-S., S. Wang, and K. Teo, Solving Hamilton—Jacobi—Bellman equations by a modified method of characteristics. Nonlinear Analysis: Theory, Methods & Applications, 2000. 40(1): p. 279-293.

[17] Martins, J.R. and A.B. Lambe, Multidisciplinary design optimization: a survey of architectures. AIAA journal, 2013. 51(9): p. 2049-2075.

[18] Alexandrov, N.M. and M.Y. Hussaini, Multidisciplinary design optimization: State of the art. Vol. 80. 1997: SIAM.

[19] Cramer, E.J., et al., Problem formulation for multidisciplinary optimization. SIAM Journal on Optimization, 1994. 4(4): p. 754-776.

[20] Deb, K., Current trends in evolutionary multi-objective optimization. International Journal for Simulation and Multidisciplinary Design Optimization, 2007. 1(1): p. 1-8.

[21] Alexandrov, N.M. and R.M. Lewis, Analytical and computational aspects of collaborative optimization for multidisciplinary design. AIAA journal, 2002. 40(2): p. 301-309.

[22] Balling, R.J. and J. Sobieszczanski-Sobieski, Optimization of coupled systems-a critical overview of approaches. AIAA journal, 1996. 34(1): p. 6-17.

[23] Keesman, K.J., System Identification: an Introduction. Advanced Textbooks in Control and Signal Processing, ed. M.J. Grimble and M.A. Johnson. 2011, London: Springer-Verlag.

[24] Nguyen, T. and S. Mukhopadhyay, Identification and Optimal Control of Large-scale System Using Selective Decentralization, in IEEE International Conference on Systems. Man and Cybernetics. 2016: Budapest.

[25] Funahashi, K.-I., On the approximate realization of continuous mappings by neural networks. Neural networks, 1989. 2(3): p. 183-192.

[26] Miller, W.T., P.J. Werbos, and R.S. Sutton, Neural networks for control. 1995: MIT press.

[27] Rumelhart, D.E., G.E. Hinton, and R.J. Williams, Learning representations by back-propagating errors. Cognitive modeling, 1988. 5(3): p. 1.

[28] Nguyen, T. and S. Mukhopadhyay, Selectively Decentralized Q-Learning, in IEEE International Conference on Systems, Man, and Cybernetics. 2017: Bannf, Canada.

[29] Bishop, C.M., Pattern Recognition. Machine Learning, 2006: p. 537-541.

[30] Russell;, S. and P. Norvig, Artificial Intelligence A Modern Approach. 3rd ed. 2010, New Jersey: Prentice Hall.

[31] Mathwork, INC. Train: Train Neural Network, retrieved from https://www.mathworks.com/help/nnet/ref/train.html, in August 2017.

[32] Wei, Q., D. Liu, and H. Lin, Value iteration adaptive dynamic programming for optimal control of discrete-time nonlinear systems. IEEE Transactions on cybernetics, 2016. 46(3): p. 840-853.

[33] Wei, Q., et al., Discrete-time local value iteration adaptive dynamic programming: Convergence analysis. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017.

[34] Kiumarsi, B., et al., Optimal tracking control of unknown discrete-time linear systems using input-output measured data. IEEE transactions on cybernetics, 2015. 45(12): p. 2770-2779.

[35] Zhang, Y., et al., Data-driven design of two-degree-of-freedom controllers using reinforcement learning techniques. IET Control Theory & Applications, 2015. 9(7): p. 1011-1021.

[36] Radac, M.-B., R.-E. Precup, and R.-C. Roman, Model-Free control performance improvement using virtual reference feedback tuning and reinforcement Q-learning. International Journal of Systems Science, 2017. 48(5): p. 1071-1083.